Saimaan ammattikorkeakoulu
Tietotekniikka Lappeenranta
Organisaation IT-palvelut

Toni Pelkola

# Googlen puheentunnistus Android-laitteilla

Opinnäytetyö 2014

## Tiivistelmä

Toni Pelkola
Googlen puheentunnistus Android-laitteilla, 37 sivua, 1 liite
Saimaan ammattikorkeakoulu
Tekniikka Lappeenranta
Tietotekniikka
Organisaation IT-palvelut
Opinnäytetyö 2014
Ohjaaja: opettaja Utti Yrjö, Saimaan ammattikorkeakoulu

Tämän opinnäytetyön tuloksena syntyi Android-käyttöliittymälle tehty ohjelma, jonka keskeisin toiminto on Googlen puheentunnistus. Googlen puheentunnistusta ja kosketusnäytön painalluksia käyttämällä ohjelmalla voi lähettää tietoa ulkoiseen tietokantaan tai lähettää tekstiviestejä.

Android-ohjelma on ohjelmoitu ymmärtämään neljää eri äänikomentoa: *apua*, *ajoin*, *tankkasin* ja *tekstiviesti*. *Apua*-komennolla ohjelma lähettää valmiiksi määriteltyyn puhelinnumeroon tekstiviestin ”apua apua apua”. *Ajoin*-komennolla käyttäjä voi lähettää ajopäiväkirjamerkinnän ulkoiseen tietokantaan, ja *tankkasin*-komennolla käyttäjä voi kirjata yhden tankkauskerran ulkoiseen tietokantaan. *Tekstiviesti*-komennolla ohjelman avulla voidaan lähettää tekstiviestejä, käyttäjän valitsemaan puhelinnumeroon. Kaikki ohjelman toiminnot käyttävät hyväkseen Googlen puheentunnistusta, pääasiassa tietojen syöttämiseen.

Opinnäytetyön ohjelman ohjelmointiympäristönä toimi Eclipse, ohjelmointikielinä Java ja PHP ja merkintäkielenä XML. Javan avulla tehtiin ohjelman sisäinen ohjelmointi, poislukien käyttöliittymä ja Android Manifest, joiden tekemiseen käytettiin XML-merkintäkieltä. PHP:ta käytettiin palvelimen PHP-tiedostojen ohjelmointiin. Palvelina toimi XAMPP-palvelinpaketin mukana tullut Apache HTTP Server.

Asiasanat: tietotekniikka, Android, Google, puheentunnistus, tietokanta, tekstiviesti, Java, PHP

## Abstract

Toni Pelkola
Google's speech recognition on Android devices, 37 pages, 1 appendix
Saimaa University of Applied Sciences
Technology Lappeenranta
Degree Programme in Information Technology
IT-services in Organisation
Bachelor´s Thesis 2014
Instructor: Senior Lecturer Yrjö Utti, Saimaa University of Applied Sciences

The result of this thesis was an Android application. The central focus of this application was Google's speech recognition. With the help of Google's speech recognition and touchscreen functions the user can send information to an external database or send text messages.

The Android application recognizes four different voice commands which are in English: *help*, *I drove*, *I fuelled* and *text message*. By saying the command *help* the application will send a text message to a predetermined phone number which says "Help, help, help". *I drove* command opens a new window which allows sending of data about the driver's log book to an external database. The command *I fuelled* also opens a new window but it allows the user to register one fuelling time to a database. And the last command *text message* enables the user to send text messages by speaking. All these functions make use of Google's speech recognition usually to input data to the application.

The programming environment was a program called Eclipse. Programming languages were Java, PHP and XML. Java was used for everything inside Eclipse excluding the user interface and Android Manifest which were made using XML. PHP was used to program PHP files which were used inside the server. The used server was XAMPP's Apache HTTP Server.

Keywords: information technology, Android, Google, speech recognition, database, text message, Java, PHP

## Sisältö

## Sanasto

| | |
|---|---|
| Aktiviteetti | Yksittäinen keskitetty asia, jonka käyttäjä voi tehdä (1). |
| Android Manifest | Android-sovellukset kuvataan Android Manifestissa. Tiedostossa kerrotaan sovelluksen käyttämät aktiviteetit, palvelut ynnä muuta sellaista. Myös tässä määritellään ohjelman tarvitseman oikeudet, kuten lupa internetin käyttämiseen (2). |
| Apache HTTP Server | Avoimen lähdekoodiin perustuva HTTP-palvelinohjelma (3). |
| Apple | Yhdysvaltalainen suuryritys, joka suunnittelee, kehittää ja myy kulutuselektroniikkaa, ohjelmistoja ja tietokoneita (4). |
| AsyncTask | Abstrakti luokka (luokka on abstrakti, jos se ei toteuta kaikkia metodeitaan), joka mahdollistaa suorittamaan operaatioita taustalla ja näyttämään tulokset käyttöliittymän puolella (5). |
| AUDREY | AUDREY oli yksi ensimmäisistä ihmisen puhetta tunnistavista laitteista (1952). AUDREY tulee sanoista Automatic Digit Recognizer (6). |
| DARPA | Defense Advanced Research Project Agency on Yhdysvaltojen puolustusministeriön alla toimiva tutkimusryhmä (7). |
| doInBackGround | Yksi AsyncTaskin askeleista, jonka aikana suoritetaan pitkään pyöriviä operaatioita (8). |
| Dragon Naturally Speaking | 1997 julkaistu puheentunnistusohjelma Windows-käyttöjärjestelmälle (9). |
| Dragon System | Julkaisi Dragon Naturally Speaking puheentunnistusohjelman (10). |
| Eclipse | Avoimen lähdekoodin ohjelmointiympäristö, tukee monia eri ohjelmointikieliä esimerkiksi Java, C#, C++, PHP (11). |

| | |
|---|---|
| EditText-kenttä | Normaali tekstikenttä käyttöliittymässä, johon voi syöttää tekstiä. |
| HARPY | 1970-luvulla luotu puheentunnistusjärjestelmä. |
| HMM | Hidden Makrov model, puheentunnistuksessa käytetty tekniikka (12.) |
| HTTP | HTTP on lyhenne sanoista Hypertext Transfer Protocol, joka on protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon. |
| HTTP-Client / asiakasohjelma | Hypertext Transfer Protocol Clientin/asiakasohjelman tarkoitus on lähettää ja vastaanottaa HTTP viestejä |
| HTTP-POST / lähete | Hypertext Transfer Protocol POSTin / lähetteen avulla lähetetään tietoa www-palvelimelle. |
| HTTP-Request / pyyntö | Esimerkiksi lähetetään palvelimelle pyyntö: ”Vastaa numerolla 6” ja palvelin vastaa sinulle antamalla takaisin tiedon ”6” (13). |
| HTTP-Response / vastaus | Hypertext Transfer Protocol Response / vastaus on WWW-palvelimen vastaus HTTP-Requestiin / pyyntöön. |
| IBM | International Business Machines, suurtietokoneiden ja raskaiden palvelimien valmistaja. |
| If-lause | If-lauseessa testataan ehto ja mikäli se on tosi, suoritetaan lauseen sisältävä koodi (14). |
| Int / Integer | Kokonaisluku. |
| Intentti | Sisältää suoritettavan toiminnon abstraktin kuvauksen (15). |
| iOS | Applen kehittämä käyttöjärjestelmä, joka on käytössä Applen iPhone-, iPod touch-, iPad- ja Apple TV–laitteissa. |
| IPv4-osoite | Internet Protocol version 4 on TCP/IP-mallin Internet-kerroksen protokolla, joka huolehtii IP-tietoliikennepakettien |

| | |
|---|---|
| | toimittamisesta perille pakettikytkentäisessä Internet-verkossa (16). |
| Java | Ohjelmointikieli. |
| Java-luokka | Määrittelee jonkin toiminnollisuuden, määrittelyn perusteella voidaan luoda luokan ilmentymiä, olioita, jotka toteuttavat tuon toiminnollisuuden. |
| JSON | JavaScript Object Notation, avoimen standardin tiedostomuoto tiedonvälitykseen. |
| Lähiverkko | LAN eli Local Area Network. mikä on rajoitetulla alueella toimiva tietoliikenneverkko. |
| Mac OS X | Applen kehittämä käyttöjärjestelmä Macintosh-tietokoneisiin. |
| Method / Metodi | Määritellyn asian tekevä ohjelman osa (17). |
| Microsoft | Yhdysvaltalainen ohjelmistoalan yritys. |
| MySQL | Relaatiotietokantaohjelmisto, tulee sanoista My (minun) ja SQL (Structured Query Language). SQL on IBM:n kehittämä kyselykieli, jolla relaatiotietokantaan voi tehdä erilaisia hakuja, muutoksia ja lisäyksiä (18). |
| mysql_query | Uniikki kysely aktiivisella tietokannalle. Esimerkiksi sen avulla voidaan muokata, lisätä tai poistaa tietoa tietokannasta. |
| NetBeans | Integroitu ohjelmointiympäristö Java-, JavaScript-, PHP-, Python-, Ruby-, Groovy-, C-, C++-, Scala- ja Clojure-ohjelmointikielille. |
| OnClick-metodi | Metodi, joka suoritetaan ohjelmassa, kun painiketta painetaan. |
| Parametri | Sisältää funktiolle tai käskylle välitettävät tiedot (19). |
| PHP | Hypertext Preprocessor on ohjelmointikieli. |

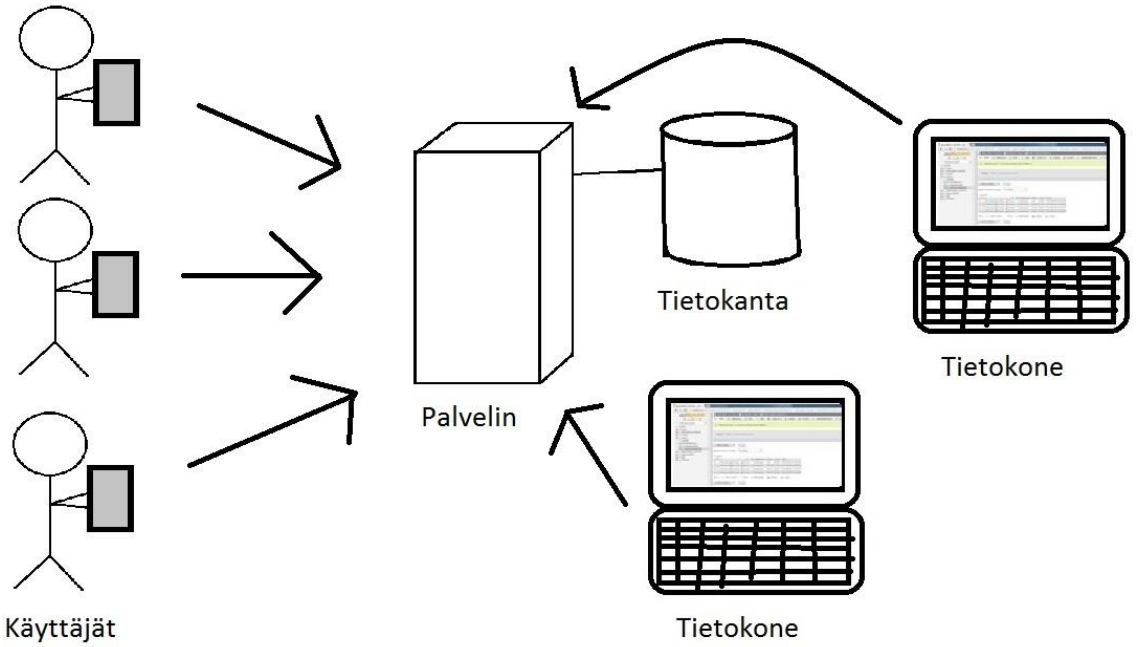
| | |
|---|---|
| phpMyAdmin | Selaimen kautta käytettävä MySQL-tietokannan hallintatyökalu. |
| SIM-kortti | Subscriber Identity Module on yksilöllinen älykortti matkapuhelinliittymän tilaajille. |
| String / merkkijono | Jono peräkkäisiä merkkejä, jotka on ohjelmoitu saman merkkijärjestelmän mukaan (20). |
| SUR | Speech Understanding Research oli 1971 Yhdysvaltojen puolustusministeriön DARPAn (Defence Advanced Reasearch Project Agency) rahoittama hanke. |
| Try- ja catch-lohko | Yritetään (try) ohjelmassa tehdä tietty asia ja (catch) määritellään, mitä tapahtuu virheen sattuessa (21). |
| Ulkoinen tietokanta | Tietokanta, joka sijaitsee ulkoisella palvelimella eikä itse laitteen sisällä. |
| URL | Uniform Resource Location on merkkijono, jolla kerrotaan tiedon paikka (22). |
| Windows Phone 8 | Microsoftin kehittämä matkaviestimiin tarkoitettu mobiilikäyttöjärjestelmä. |
| Windows Vista | Microsoft Windowsin käyttöjärjestelmä, julkaistu yleiskäyttöön vuonna 2007. |
| XAMPP | Web-palvelinpaketti. Jonka nimi tulee sen sisältävistä toiminnoista/ohjelmista: X(Cross-platform), Apache HTTP Server, MySQL, PHP, Perl(23). |
| XML | Extensible Markup Language, joka on merkintäkieli internetympäristössä. |

# 1 Johdanto

Opinnäytetyössä perehdytään siihen, miten Android-käyttöjärjestelmällä Googlen puheentunnistus toimii ohjelmointiympäristössä ja miten sitä voi käyttää hyväksi ohjelmissa, kuten tietojen lisäämisessä ulkoisiin tietokantoihin ja tekstiviestien lähettämiseen. Puheentunnistusohjelma toimii tällä hetkellä matkapuhelimissa ja taulutietokoneissa, joissa on Android-käyttöjärjestelmä. Yllämainittujen asioiden perusteella tehtiin Android-ohjelma, jonka avulla testattiin näitä ominaisuuksia. Opinnäytetyössä käytetty versio Googlen puheentunnistuksesta toimii Googlen verkkopalvelimella ja vaatii internetyhteyden toimiakseen.

Ohjelma toimii yksinkertaisesti. Kun ohjelma avataan, näkyy mobiililaitteen näytöllä yksi iso ruudun kokoinen kosketuksella painettava painike. Kun kyseistä painiketta painetaan, kosketusnäytöltä avautuu uusi pieni ikkuna, joka aktivoi Googlen puheentunnistuksen. Nyt käyttäjän pitää sanoa ohjelmalle yksi valmiiksi määritellyistä komennoista. Riippuen annetusta komennosta ohjelmalla voi tehdä neljää eri tehtävää: lähettää tekstiviestejä, lähettää avunpyynnön suoraan tekstiviestillä valmiiksi ohjelmointiympäristön puolella määriteltyyn puhelinnumeroon, kirjata polttoaineen tankkauskerran ulkoiseen tietokantaan tai kirjata ajopäiväkirjaa ulkoiseen tietokantaan. Ulkoinen tietokanta on tietokanta, joka sijaitsee ulkoisella palvelimella eikä itse laitteen sisällä.

Ohjelmaa voi samanaikaisesti käyttää monelta eri Android-laitteelta, jolloin kaikkien käyttäjien lähettämät tiedot lisätään samaan tietokantaan. Tietokantaan lisättyjä tietoja voi katsoa tietokannasta useammalta tietokoneelta (Kuva 1).




Kuva 1. Yleiskuva opinnäytetyön ohjelman toiminnasta

Opinnäytetyössä näytetään joka kohdassa ensin, miten toiminto toimii tehdyn Android-ohjelman käyttöliittymässä ja tämän jälkeen kerrotaan, mitä tarvitsee ohjelmoida, jotta toiminto saadaan toimimaan. Opinnäytetyössä kerrotaan myös, miten oma paikallinen palvelin tietokantoineen pystytetään ja miten ohjelman avulla saadaan lähetettyä tietoja kyseiseen tietokantaan PHP-ohjelmointikieltä (Hypertext Preprocessor) käyttäen. Puheentunnistuksen historiaan perehdytään lyhyesti.

# 2 Puheentunnistuksen historia 1950-luvulta nykypäivään

Yksi ensimmäisistä ihmisen puhetta tunnistavista laitteista oli AUDREY (Automatic Digit Recognizer) (1952). AUDREY tunnisti sanat 97–99 prosentin todennäköisyydellä. Ongelmana oli, että laite piti säätää puhujaa varten ja laitteeseen sanoessa piti pitää aina sanojen välissä tauko, mikä teki siitä erittäin epäkäytännöllisen.

1971 alkoi Yhdysvaltojen puolustusministeriön DARPAn (Defence Advanced Research Project Agency) rahoittama hanke SUR (Speech Understanding Research). Yksi heidän saannoksistaan oli HARPY puheentunnistusjärjestelmä. HARPY saavutti 95 prosentin virheettömyyden puheentunnistuksessa, tosin

HARPY-järjestelmällä kesti 4 minuuttia prosessoida lause, jonka sanomiseen meni 3 sekuntia aikaa.

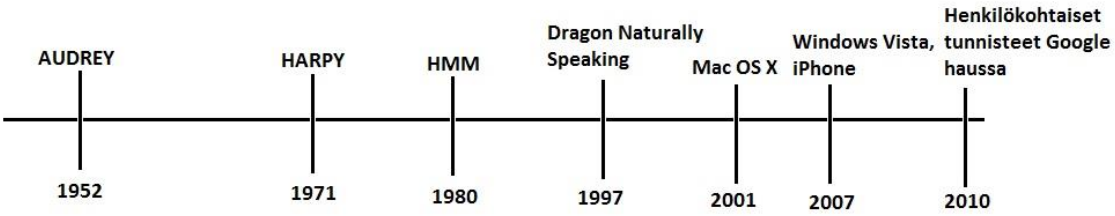
1980-luvulla alettiin hyödyntämään tekniikkaa nimeltä HMM (hidden Makrov model). Pelkkien äänikuvioiden tutkimisen sijasta HMM tarkasteli tuntemattomien äänien todennäköisyyttä olla sanoja. Tätä tekniikka käytetään nykyäänkin monissa puheentunnistuksissa. Vaikka sanastojen koot nousivat sadoista sanoista tuhansiin sanoihin, silti tauotus sanottaessa laitteille säilyi.

Heinäkuussa 1997 Dragon System julkaisi ensimmäisen yleiseen käyttöön suunnatun Dragon Naturally Speaking tietokoneohjelman kotikoneille. Ohjelma sisälsi 23000 sanan sanaston ja tarjosi jatkuvan puheentunnistuksen. IBM (International Business Machines, suurtietokoneiden ja raskaiden palvelimien valmistaja) julkaisi samankaltaisen ohjelman kuukausi Dragon Naturally Speaking tietokoneohjelman julkaisun jälkeen. Myöhemmin Microsoft (Yhdysvaltalainen ohjelmistoalan yritys) ja Apple (Yhdysvaltalainen suuryritys, joka suunnittelee, kehittää ja myy kulutuselektroniikkaa, ohjelmistoja ja tietokoneita) alkoivat tarjota äänikomennoilla toimivia tuotteita tietokoneille.

Ensimmäiset käyttöjärjestelmät, joihin oli sisään rakennettu puheentunnistus, olivat Windows Vista (Microsoft Windowsin käyttöjärjestelmä, julkaistu yleiskäyttöön vuonna 2007) ja Mac OS X (Applen kehittämä käyttöjärjestelmä Macintosh-tietokoneisiin). Puheentunnistus ja äänikomennot eivät olleet tarpeeksi tarkkoja tai helppoja käyttää, joten ne vielä jäivät näppäimistön ja hiiren jalkoihin. (24)

Googlen Voice Search-ohjelma iPhonelle oli yksi nykypuheentunnistuksen kulmakiviä, ja 2010 Google lisäsi äänihakuun ”henkilökohtaisen tunnistuksen”, jonka avulla ohjelma nauhoitti käyttäjien äänihaut ja tuotti näiden avulla tarkemman puheentunnistuksen. (25)

Puheentunnistuksen aikajana havainnollistettuna kuvallisesti (Kuva 2).

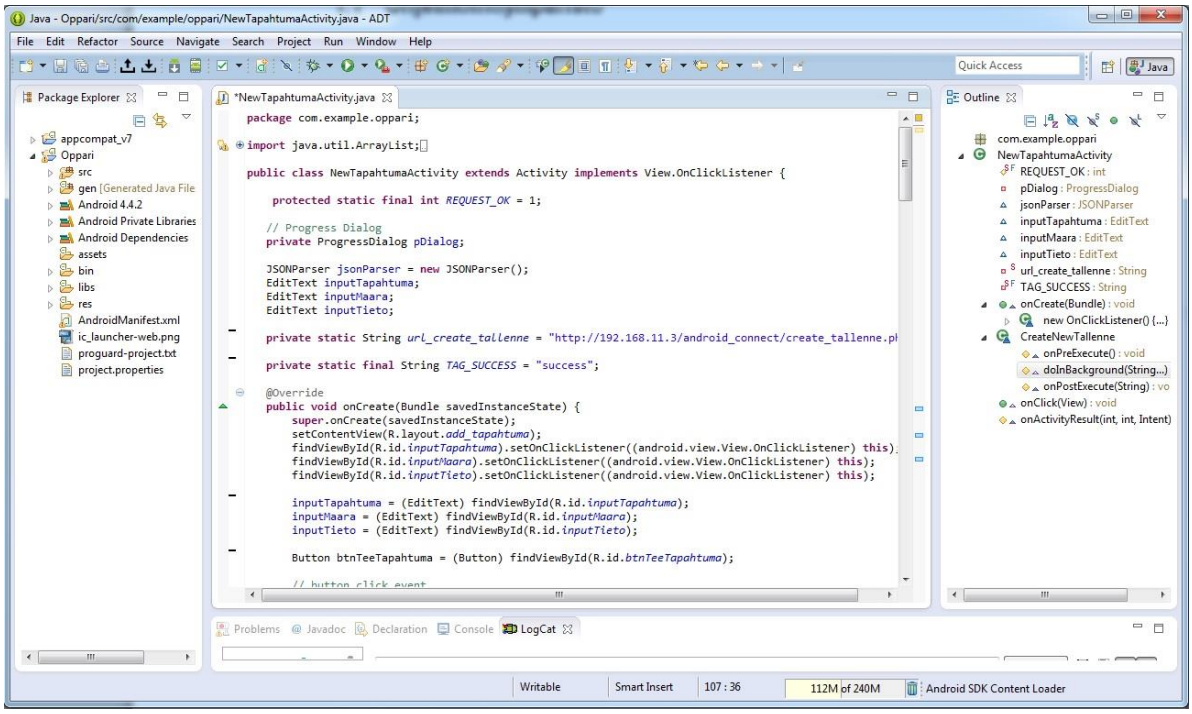



Kuva 2. Puheentunnistuksen aikajana

Nykyään erilaiset puheentunnistukset tunnistavat melkein kaikki kielet ja puheentunnistus on erittäin hyvä kilpailija perinteiselle näppäimistöllä kirjoittamiselle sekä nopeudessa että käytännöllisyydessä. Googlen puheentunnistus on palvelu, joka toimii Googlen omilla palvelimilla. Tämä mahdollistaa palvelun jatkuvan kehityksen ilman, että käyttäjien laitteille asennetaan koko ajan uudempia versioita Googlen puheentunnistuspalvelusta.

# 3 Ohjelmointiympäristö ja kielet

Opinnäytetyön tekemisessä ohjelmointiympäristönä käytettiin avoimeen lähdekoodiin perustuvaa Eclipseä (Kuva 3), joka tarjoaa työympäristön muun muassa Android- ja Java-projektien tekemistä varten. Eclipse tukee eri ohjelmointikieliä, esimerkiksi Java ja PHP. Vastaavilla ominaisuuksilla toimii myös NetBeans, joka on integroitu ohjelmointiympäristö. Eclipse valittiin NetBeansin sijaan paremman Android-laitetuen takia.

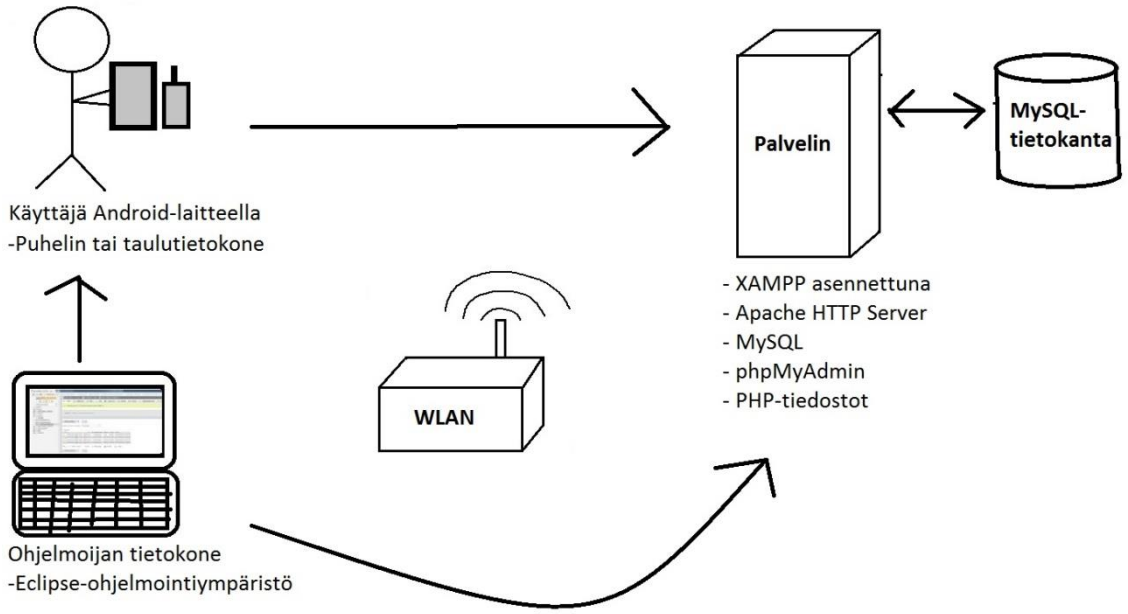



Kuva 3. Kuvankaappaus Eclipsestä

XAMPP web-palvelinpakettia käytettiin paikallisen palvelimen pystyttämiseen. XAMPP koostuu seuraavista ominaisuuksista ja sanoista: X(Cross-platform), Apache HTTP Server (avoimeen lähdekoodiin perustuva HTTP-palvelinohjelma), MySQL-relaatiotietokantaohjelmisto (My Structured Query Language), PHP ja Perl.

Näistä tässä työssä käytettiin Apache HTTP (Hypertext Transfer Protocol) Serveriä, MySQL:lää ja PHP:tä. MySQL:n avulla pidetään tietokannat kunnossa ja PHP:n avulla lähetetään Android-ohjelman lähettämät tiedot tietokantaan. Paketin mukana tuli myös phpMyAdmin, joka on selaimen kautta käytettävä MySQL-tietokannan hallintatyökalu, jonka kautta voi selata tietokantoja ja tehdä siellä muutoksia käyttöliittymän kautta ilman SQL-kyselyitä.

Seuraavan kuvan (Kuva 4) avulla saa paremman havainnollisen käsityksen palvelimen ja ohjelmistoympäristön toimintaperiaatteesta.



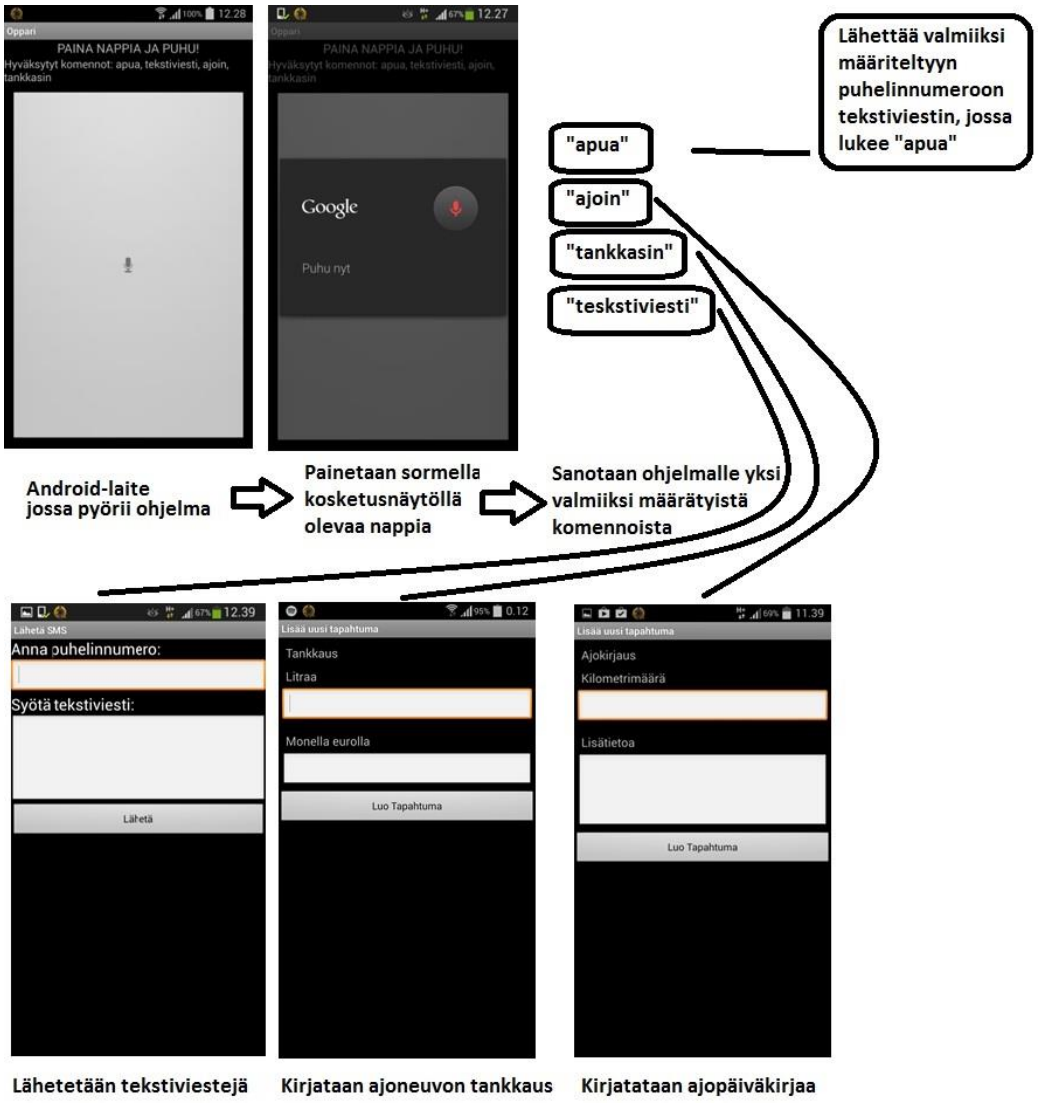

Kuva 4. Yleiskuva ohjelmointiympäristön ja palvelimen tilanteesta

Ohjelmoijan tietokoneella ohjelmoitiin Android-ohjelma Eclipse-ohjelmistoympäristön avulla, jonka kautta myös asennettiin ohjelma Android-laitteelle. Android-laite lähettää syötetyt tiedot palvelimen PHP-tiedostoihin, joiden avulla ne lisätään MySQL-tietokantaan. Tietokannan tietoja voi selata palvelimelle asennetusta phpMyAdminista tietokoneen selaimen kautta. Android-laitteen, palvelimen ja tietokoneen pitää olla kaikki samaan lähiverkkoon yhdistettynä, jotta tiedonkulku toimii.

Opinnäytetyön ohjelmaa tehtäessä käytettiin Java- ja PHP-ohjelmointikieliä, myös XML-merkintäkieltä (Extensible Markup Language). Javaa käytettiin puhelimen luokkien ohjelmointiin ohjelmointiympäristössä. PHP:lla tehtiin tarvittavat PHP-tiedostot tietokantayhteyksiä varten palvelimelle, ja XML:ää käytettiin käyttöliittymän että Android Manifestin ohjelmointiin ohjelmointiympäristössä. Android Manifest on tiedosto, jossa Android-sovellukset kuvataan, tiedostossa kerrotaan sovelluksen käyttämät aktiviteetit, palvelut ja määritellään ohjelman tarvitsemat oikeudet. Aktiviteetti on yksittäinen keskitetty asia, jonka käyttäjä voi tehdä.

# 4 Googlen puheentunnistus Android-laitteissa

Ohjelman käyttöliittymässä (Kuva 5) painetaan Android-laitteen kosketusnäytöltä isoa harmaata painiketta, jolloin Googlen puheentunnistus aktivoituu ja avaa uuden pienen ikkunan. Nyt käyttäjän pitää sanoa laitteelle jokin seuraavista komennoista: *apua, ajoin, tankkasin* tai *tekstiviesti*. Riippuen sanotusta komennosta tapahtuu sille määritelty tapahtuma ohjelmassa. *Apua* lähettää valmiiksi määriteltyyn puhelinnumeroon tekstiviestin, jossa lukee ”Apua, apua apua*”*. *Ajoin*-komento puolestaan avaa ikkunan, jonka kautta kirjataan ajopäiväkirjaa. *Tankkasin*-komento avaa ikkunan, jonka avulla kirjataan yksi tankkauskerta. *Tekstiviesti*-komennolla aukeaa ikkuna, jonka kautta voi lähettää tekstiviestejä käyttäen puhetta.



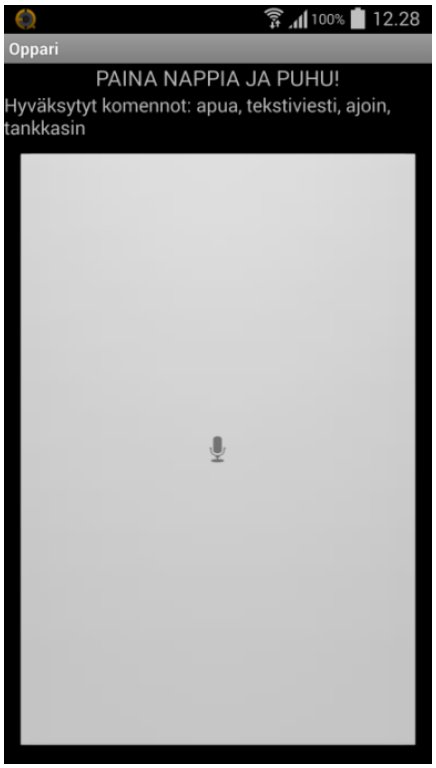

Kuva 5. Havainnollinen kuva ohjelman toimintaperiaatteesta

Yksinkertaisuudessaan Googlen pilvipalvelu yrittää ymmärtää sanotusta lauseesta konsonantit ja vokaalit. Seuraavaksi se käyttää saatuja tietoja tehdessään älykkäitä arvauksia siitä, mikä sanottu sana voisi olla.

### 4.1 Googlen puheentunnistus käyttöliittymän puolella

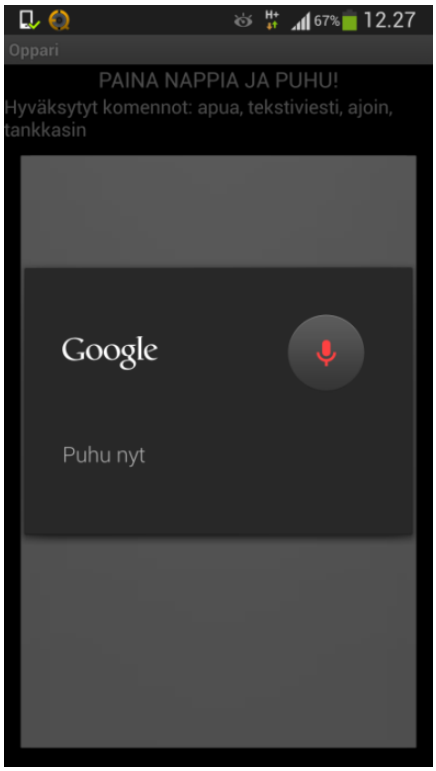
Opinnäytetyössä Googlen puheentunnistusta käytetään kahteen eri tarkoitukseen: alkukomennon antamiseen ja tietojen syöttämiseen.

Googlen puheentunnistus aktivoituu, kun Android-laitteen kosketusnäytöltä painetaan isoa harmaata ruudun kokoista painiketta sormella (Kuva 6).




Kuva 6. Opinnäytetyön ohjelman aloitusnäkymä

Kun ohjelmassa on Googlen puheentunnistus aktivoitunut (Kuva 7) ja ohjelmalle on sanottu haluttu komento, ohjelma muuttaa sen sisäisen merkkijonon siihen, mitä sille on sanottu. Jos komento on jokin seuraavista: *apu*a, *tekstiviesti*, *ajoin* tai *tankkasin*, tapahtuu sille komennolle määritelty toiminto. Mikäli ohjelmalle sanotaan tuntematoman komento, annetaan käyttäjälle seuraava viesti: ”komentoa ei ole tunnistettu”.




Kuva 7. Näkymä, kun Googlen puheentunnistus on aktivoitunut

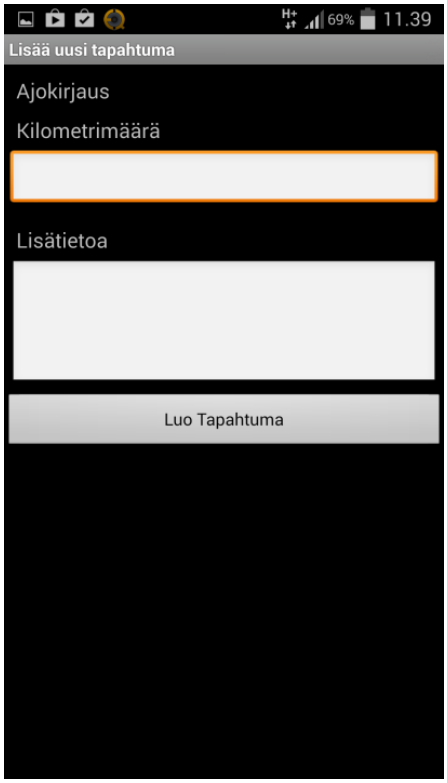
Alla olevassa ohjelman osassa on esimerkki siitä, mitä tapahtuu, kun ohjelma tunnistaa komennoksi *tankkasin*. Aluksi määritellään haluttu luokka-aktiviteetti, joka tullaan aloittamaan ja kun se aktivoidaan.

```
if(view.getText().equals("tankkasin"))
{
Intent i = new Intent(getApplicationContext(), NewTapahtumaAc-
tivity.class);
startActivity(i);
}
```

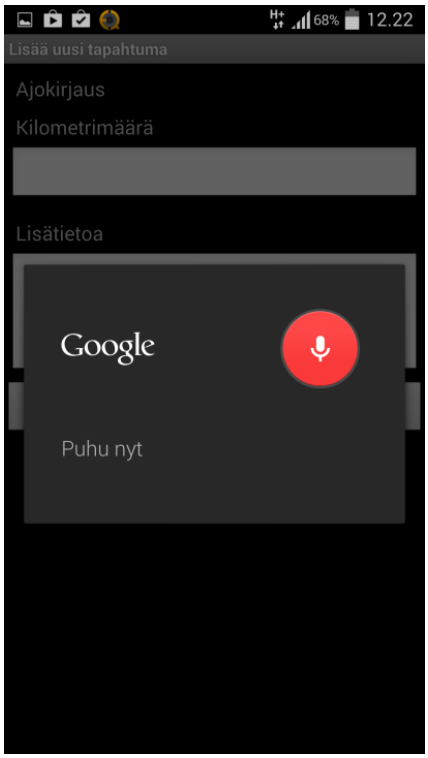
Toinen tapaus, jossa puheentunnistusta käytetään, on tietojen syöttäminen, joka tapahtuu ajon ja tankkauksen kirjaamisessa.

Kun esimerkiksi *ajokirjausikkuna*-ikkuna on auki (Kuva 8), painetaan kilometrimäärän alla olevaa tekstikenttää, silloin aktivoituu Googlen puheentunnistus (Kuva 9). Tämän jälkeen sanotaan haluttu kilometrimäärä (tässä tapauksessa *35* (kolmekymmentäviisi)). Ohjelma tunnistaa sanotut numerot ja lisää arvot tekstikenttään. Samalla tapaa toimii lisätietoa-kohta. Kun lisätietoa-tekstin alta painetaan tekstikenttää, aktivoituu Googlen

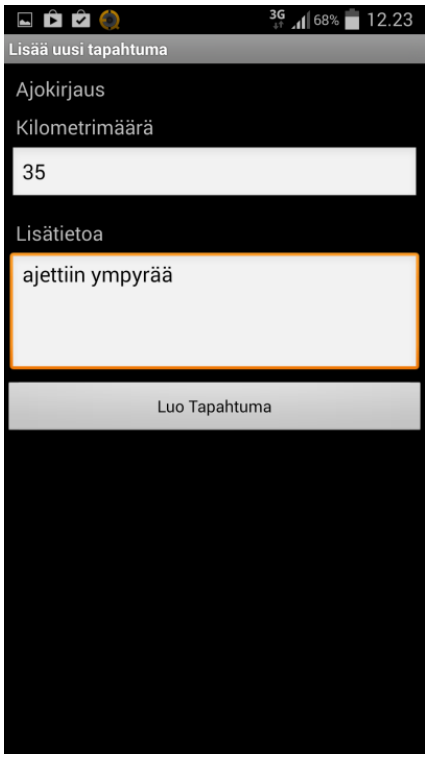
puheentunnistus, ja siihen sanottaessa ääneen haluttu sana (tässä tapauksessa ”ajettiin ympyrää”) se ilmestyy tekstikenttään (Kuva 10).




Kuva 8. Näkymä *ajokirjauksesta*




Kuva 9. Näkymä tietojen lisäämisestä *ajokirjaukseen*




Kuva 10. Ruudunkaappaus tietojen lisäämisen jälkeen *ajokirjauksesta*

### 4.2 Googlen puheentunnistuksen kutsuminen ohjelmassa

Alapuolella näytetään esimerkki sanotun sanan lisäämisestä tekstikenttään:

```
@Override
public void onClick(View v)
{
Intent i = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
i.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, "en-US");
    try {
        startActivityForResult(i, REQUEST_OK);
    } catch (Exception e) {
           Toast.makeText(this, "Virhe käynnistäessä puheen
tunnistusta.", Toast.LENGTH_LONG).show();
    }
   }
```

Aluksi ohjelmointiympäristössä Javalla luodaan metodi *onClick*. Seuraavaksi määritellään *onClickin* sisälle toiminto *ACTION_RECOGNIZE_SPEECH*, mikä mahdollistaa puheentunnistuksen kutsumisen. Myös sille lisätään uusi kieli käyttämällä komentoa *putExtra (RecognizerIntent. EXTRA_LANGUAGE_MODEL, ”en-US”);*. Vaikka ohjelmointiympäristössä

kieleksi on määritelty *en-US*, joka tarkoittaa amerikanenglantia, tunnistaa puheentunnistus siitä huolimatta myös esimerkiksi suomen kielen.

*Try*- ja *catch*-lohkoihin (yritetään (*try*) ohjelmassa tehdään tietty asia ja (*catch*) määritellään mitä tapahtuu virheen sattuessa) lisätään itse puheentunnistuksen aloitus ja määritellään, mitä tehdään, jos puheentunnistusta käynnistäessä tapahtuu virhe. Puheentunnistusaktiviteettia kutsutaan *startActivityForResult*-komennolla, komennon sisään lisätään myös kohta *REQUEST_OK*, jotta ohjelma tietäisi tietojen lähteneen eteenpäin Googlen palvelimelle. *Catch*-lohkoon lisätään *Toast.makeText*, joka on ponnahdusikkuna, jossa kerrotaan puheentunnistuksen aktivoinnin aikana tapahtuneesta virheestä. Tällä tekniikalla voidaan välttää ohjelman sulkeutuminen ennenaikaisesti, jos ohjelmaa suorittaessa syntyy virheitä.

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
Intent data)
{
super.onActivityResult(requestCode, resultCode, data);
if (requestCode==REQUEST_OK  && resultCode==RESULT_OK) {
        ArrayList<String> thingsYouSaid = da-
ta.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
                    ((EditText) findViewBy-
Id(R.id.EditText1)).setText(thingsYouSaid.get(0));
  }
}
```

Seuraavaksi tehdään metodi *onActivityResult* (JOKA suoritetaan, kun Googlen palvelimelta ollaan hakemassa tiedot takaisin), joka pitää sisällään seuraavat *intit* (integer, kokonaisluvut) ja *Intentit* (sisältävät suoritettavien toimintojen abstraktit kuvaukset): *int requestCode*, *int resultCode* ja *Intent data*. Mistä *requestCode* kertoo, onko puheentunnistuksen pyyntö lähetetty palvelimelle (arvo on joko *REQUEST_OK* tai se ei ole mitään). *ResultCode* puolestaan on palvelimen palauttama tieto siitä, onko puheentunnistus onnistunut (palauttaa arvon *RESULT_OK*, jos toimenpide on onnistunut). Lopuksi käsitellään *Intent data*, joka on palvelimen palauttama arvo siitä, joksi puheentunnistus on sanotun tekstin tunnistanut.

Aktiviteetin tulokset haetaan ensin komennolla *super.OnActivityResult(requestCode, resultCode, data);*. Tämän jälkeen tarkastetaan *if*-lauseella, ovatko *requestCode* ja *resultCode* kummatkin *OK*, eli onko puheentunnistuksen pyyntö lähtenyt palvelimelle ja onko puheentunnistus onnistunut. *If*-lauseessa testataan ehto, ja mikäli se on tosi, suoritetaan lauseen sisälle ohjelmoidut asiat.

Kun *if*-lauseesta päästään ohi, muutetaan palvelimelta tullut data sellaiseen muotoon, että sitä voidaan käyttää hyväksi. Luodaan *ArrayList<String> thingsYouSaid*, joka on lista merkkijonoja. Tähän listaan lisätään puheentunnistuksen kautta tullut tieto käyttäen seuraavaa ohjelmakoodia: *data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);.*

Jotta tulos saataisiin näkymään vielä ohjelman puolella, lisätään se *EditText*-kenttään (normaali tekstikenttä käyttöliittymässä, johon voi syöttää tekstiä) seuraavasti: *.setText(thingsYouSaid.get(0));*, jossa *setText*-komennolla listään *EditText*-kenttään teksti, joka on kirjoitettu sen sulkeisiin.

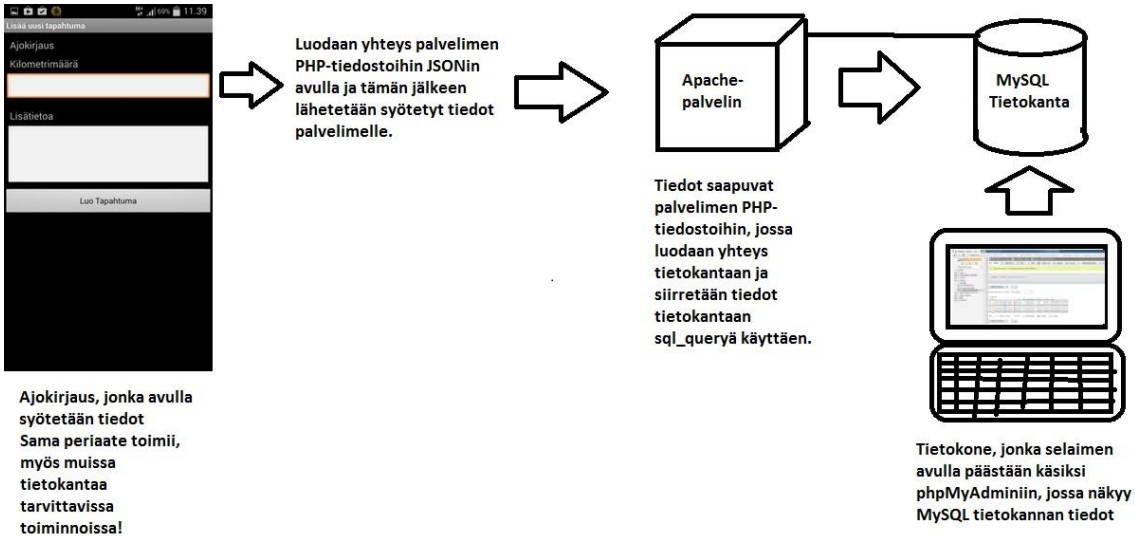
Jotta puheentunnistusaktiviteetti toimisi, pitää Android Manifestiin lisätä seuraava rivi: *<uses-permission android:name="android.permission.INTERNET"/>*. Tämä antaa ohjelmalle luvan käyttää internetiä puheentunnistuksen aktivointia varten.

# 5 Palvelin ja tietokanta

Ohjelma lähettää ulkoiseen tietokantaan verkon välityksellä tietoa, sen mukaan, mitä käyttäjä on puhelimelle sanonut. Esimerkiksi jos ohjelmalla kirjaa auton tankkauksen, tietokantaan lisätään tankkauksesta seuraavat tiedot: litramäärä, hinta euroina ja aika jolloin tankkaus on tehty.

Tietokantayhteyden luominen alkaa heti siitä kohtaa, kun käyttäjä painaa *Lähetä Tiedot*-painiketta ohjelmassa (Kuva 11)(tai kun ohjelmalla lähetetään avunpyyntö *apua*-komennolla). Aluksi ohjelma muuntaa JSONin (JavaScript Object Notation, avoimen standardin tiedostomuoto tiedonvälitykseen) avulla syötetyt tiedot sellaiseen muotoon, että ulkoisen tietokannan palvelimen PHP-

koodi tunnistaa annetut tiedot. Tämän jälkeen ohjelma lähettää muunnetun tiedon internetin välityksellä ulkoisen tietokannan palvelimen PHP-tiedostoon, joka tunnistaa annetun tiedon ja lisää sen MySQL-tietokantaan. Tämän toimenpiteen jälkeen on ohjelmalle annettu tieto valmiiksi katseltavissa palvelimella olevasta phpMyAdminista, joka on selaimen kautta käytettävä MySQL-tietokannan hallintatyökalu.




Kuva 11. Ulkoisen tietokannan toimintaperiaate

Ohjelmaa testatessa palvelimena toimi paikallinen XAMPP-ohjelmalla luotu Apache-palvelin, XAMPPia käyttäen luotiin myös MySQL-tietokannat ja samalla saatiin palvelimelle PHP-tuki.

## 5.1 JSON

JSON on yksinkertainen avoimen standardin tiedostomuoto tiedonvälitykseen. Opinnäytetyössä JSON ilmenee Java-luokkana (määrittelee jonkin toiminnollisuuden, määrittelyn perusteella voidaan luoda luokan ilmentymiä, olioita, jotka toteuttavat tuon toiminnollisuuden) ohjelmointiympäristön puolella, jonka avulla voidaan lähettää ohjelman keräämät tiedot palvelimen PHP-tiedostojen kautta ulkoiseen tietokantaan.

Tietojen lähettämiseen tarvitaan kaksi  asiaa: PHP-tiedoston URL-osoitteen ja parametrit (sisältävät funktioille tai käskyille välitettävät tiedot), jotka lähetetään

eteenpäin. URL (Uniform Resource Location) tarkoittaa merkkijonoa, jolla kerrotaan tiedon paikka ohjelmointiympäristössä.

URL-osoite voidaan määrittää ohjelmointiympäristössä seuraavalla tavalla:

```
private static String url_create_tallenne = "http://192.168.43.240/android_connect/create_tankkaus.php";
```

Tämä on URL-osoite on tie tankkaamisen tietojen lähettämistä varten olevaan PHP-tiedostoon. Muiden tietojen lähettämistä varten on omat URL-osoitteet.

Sitten rakennetaan parametrit, eli yhdistetään kaikki käyttöliittymään syötetyt tiedot samaan taulukkoon:

```
List<NameValuePair> params = new ArrayList<NameValuePair>();
params.add(new BasicNameValuePair("tapahtuma", tapahtuma));
params.add(new BasicNameValuePair("maara", maara));
params.add(new BasicNameValuePair("euroa", euroa));
```

Lopuksi tehdään JSON-luokalle HTTP-pyyntö, joka sisältää tankkauksen PHP-tiedoston URL-osoitteen.

Tämän jälkeen lähetetään tiedot palvelimen PHP-tiedostoihin:

```
DefaultHttpClient httpClient = new DefaultHttpClient();
HttpPost httpPost = new HttpPost(URL);
httpPost.setEntity(new UrlEncodedFormEntity(postparams));
HttpResponse httpResponse = httpClient.execute(httpPost);
```

Ennen HTTP-pyynnön tekemistä pitää kaikki lähetettävä data koodata, jotta merkkijonon tiedot saadaan muutettua URL-osoitteelle tarkoitettuun muotoon (13). Muuntamisen jälkeen suoritetaan tietojen lähettäminen käyttämällä HTTP-clienttiä (HTTP-Clientin tarkoitus on lähettää ja vastaanottaa HTTP-viestejä). Lopuksi HTTP-Responsen avulla haetaan vastaus palvelimelta: toimiko tietojen lähettäminen tietokantaan.

## 5.2 PHP ja mysql_query

PHP:n avulla pystytään luomaan yhteys ulkoiseen tietokantaan ja sitä kautta lähettämään ohjelmasta tietoa ulkoiseen tietokantaan.

Kun *Lähetä Tiedot*-painiketta painetaan, kirjautuu ohjelma PHP:n avulla ulkoiseen tietokantaan sisään käyttämällä luotuja tunnuksia ja sille annettua salasanaa (Kuva 12), jotta päästetään lisäämään tietokantaan tietoja. Ilman käyttäjänimeä ja salasanaa ei tietokannan tietoihin pääse käsiksi, eli tietojen tallennus tietokantaan on mahdotonta. Yhteys tietokantaan tehdään seuraavan PHP-ohjelmoinnin avulla:

```
mysql_connect(DB_SERVER, DB_USER, DB_PASSWORD);
```

DB_SERVER tarkoittaa tietokannan palvelimen IP-osoitetta, DB_USER on tietokannan käyttäjänimi ja DB_PASSWORD salasana tietokantaan. Itse tietokanta valitaan PHP:ssa seuraavasti:

```
mysql_select_db(DB_DATABASE);
```

jossa DB_DATABASE kohtaan laitetaan tietokannan nimi, johon tiedot tallennetaan.

```
1  <?php
2
3  define('DB_USER', "root"); // tietokannan käyttäjänimi
4  define('DB_PASSWORD', ""); // tietokannan salasana
5  define('DB_DATABASE', "oppari"); // tietokannan nimi
6  define('DB_SERVER', "localhost"); // tietokannan serveri
7  ?>
```

Kuva 12. Kuvankaappaus db_config.php tiedostosta

Ohjelma lähettää syötetyt tiedot PHP:n avulla tietokantaan seuraavasti ohjelmistoympäristössä:

```
mysql_query("INSERT INTO tankkaustallenteet(tapahtuma, litraa, euroa) VALUES('$tapahtuma', '$litraa', '$euroa')");
```

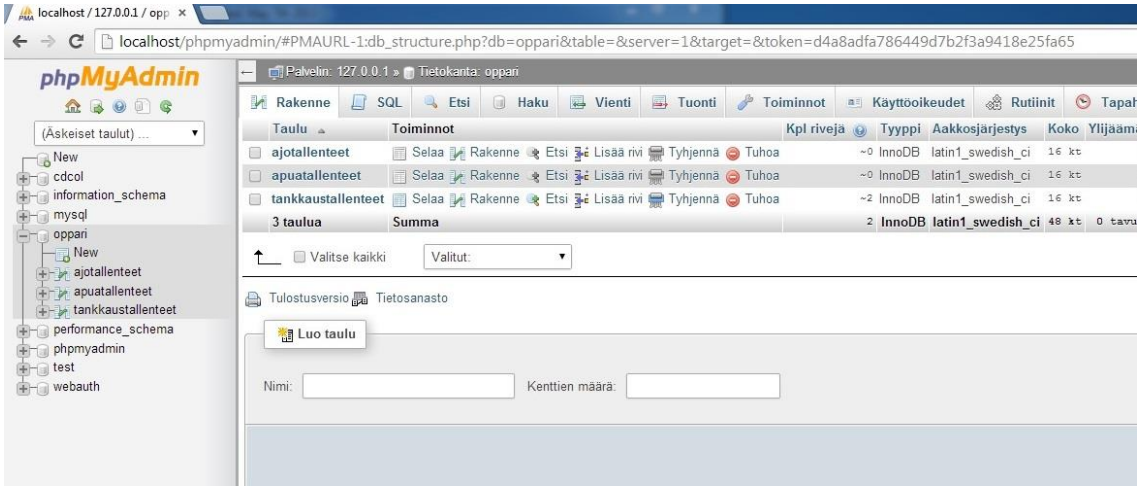
Mysql_query on yksittäinen kysely silla hetkellä aktiiviselle tietokannalle. Kyselyiden avulla voidaan esimerkiksi syöttää tietokantaan tietoa, hakea sieltä

tietoa tai muokata sitä. Yllämainitun mysql_queryn avulla lisätään tietokantatauluun nimeltä *tankkaustallenteet* seuraavat tiedot: tapahtuma eli tankkaus, litramäärä, joka on syötetty ohjelmalle, ja euromäärä, joka on ohjelmalle annettu. Kun PHP-koodi on ajettu loppuun ja tiedot lähetetty tietokantaan, näkyvät syötetyt tiedot selaimella phpMyAdminista.

## 5.3 phpMyAdmin

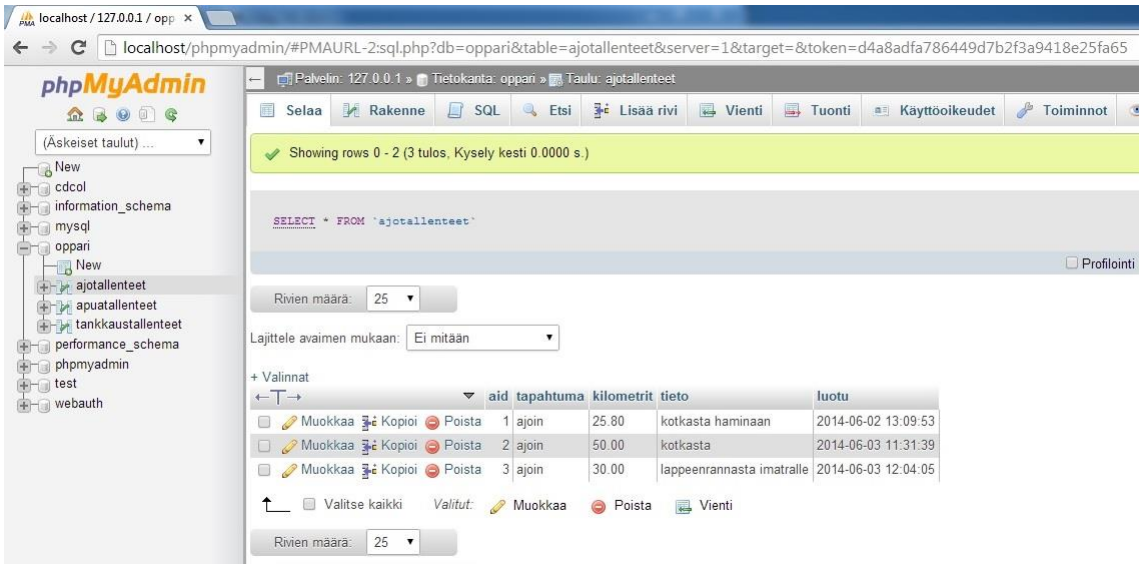
PhpMyAdmin on ilmainen PHP:lla kirjoitettu työkalu, jolla on tarkoitus hallita MySQL-tietokantoja selainta käyttämällä.

Heti kun phpMyAdmin on avattu selaimella, valitaan haluttu tietokanta (tässä tapauksessa *oppari*-tietokanta) ja sivu avautuu selaimelle, jossa näkee kaikkien tietokantataulujen nimet, jotka löytyvät tietokannasta (Kuva 13).




Kuva 13. Kuvankaappaus phpMyAdminin tietokannan tauluista

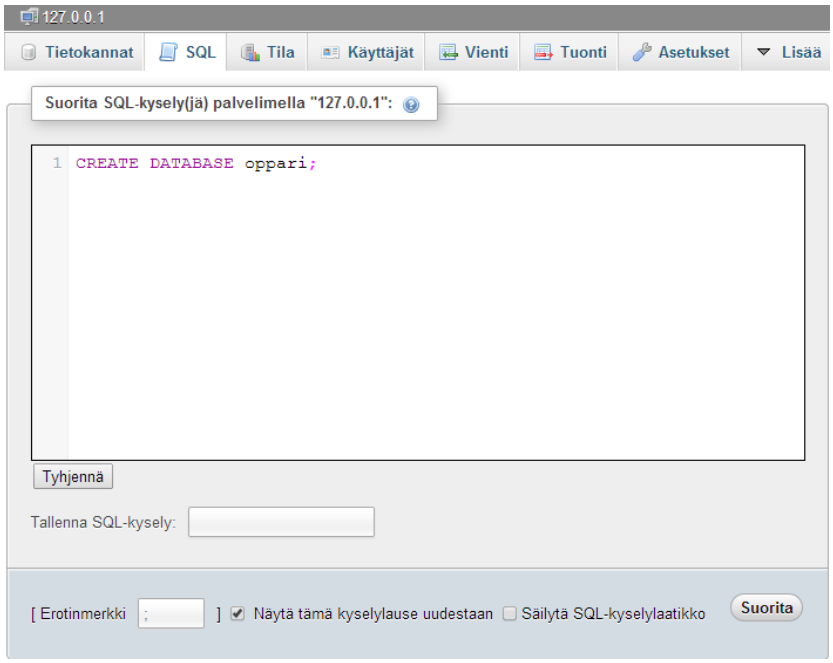
Kun tietokantataulua nimeltä *ajotallenteet* painetaan, avautuu sivu, jossa näkyy kaikki tallennetut tiedot, joita ohjelmalla on syötetty tietokantaan (Kuva 14).




Kuva 14. Kuvankaappaus phpMyAdminin *ajotallenteet*-taulusta

Kuvasta 14. nähdään, että *ajotallenteet*-tauluun on lisätty tietoa kolme kertaa, koska sarakkeita on kolme. PhpMyAdminin avulla voi myös poistaa tai muokata syötettyjä tietoja.

Tietokanta ja taulut luotiin käyttämällä phpMyAdminista löytyvää SQL-toimintoa (Kuva 15).

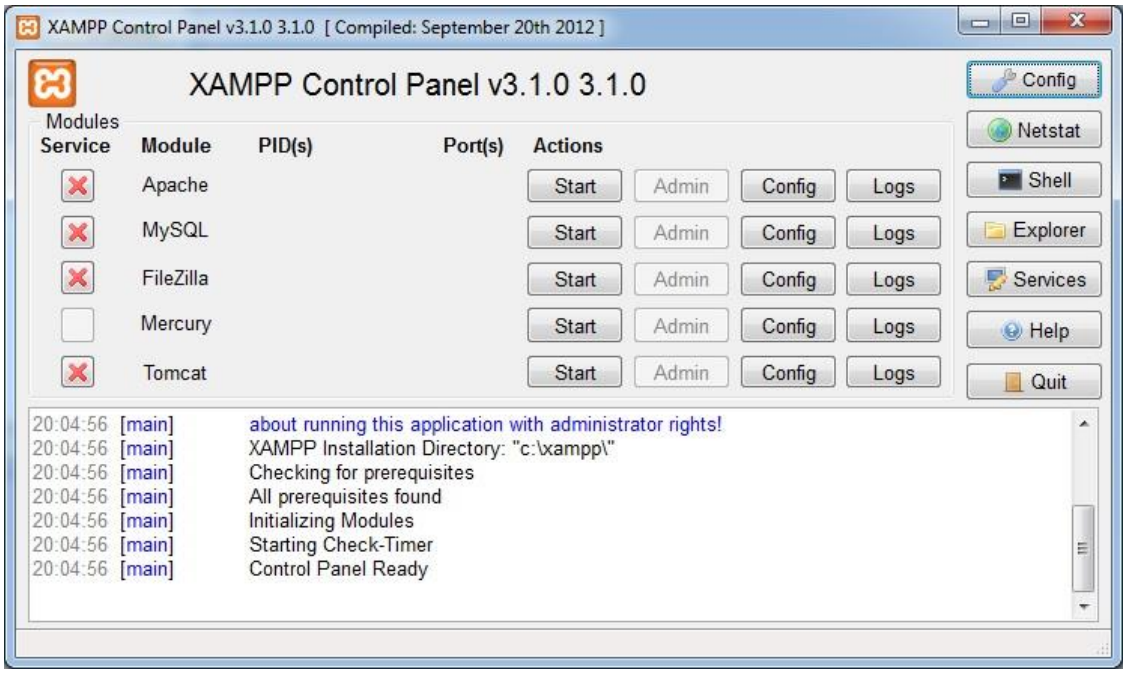



Kuva 15. Tietokannan luominen phpMyAdminissa SQL:n avulla

PhpMyAdminin etusivulta painetaan SQL-saraketta, jotta phpMyAdminiin saadaan syötettyä SQL-lauseita. Sitten vain kirjoitetaan tekstikenttään haluttu SQL-lause, tässä tapauksessa *CREATE DATABASE oppari;* , jonka avulla luodaan tietokanta nimeltä *oppari*.

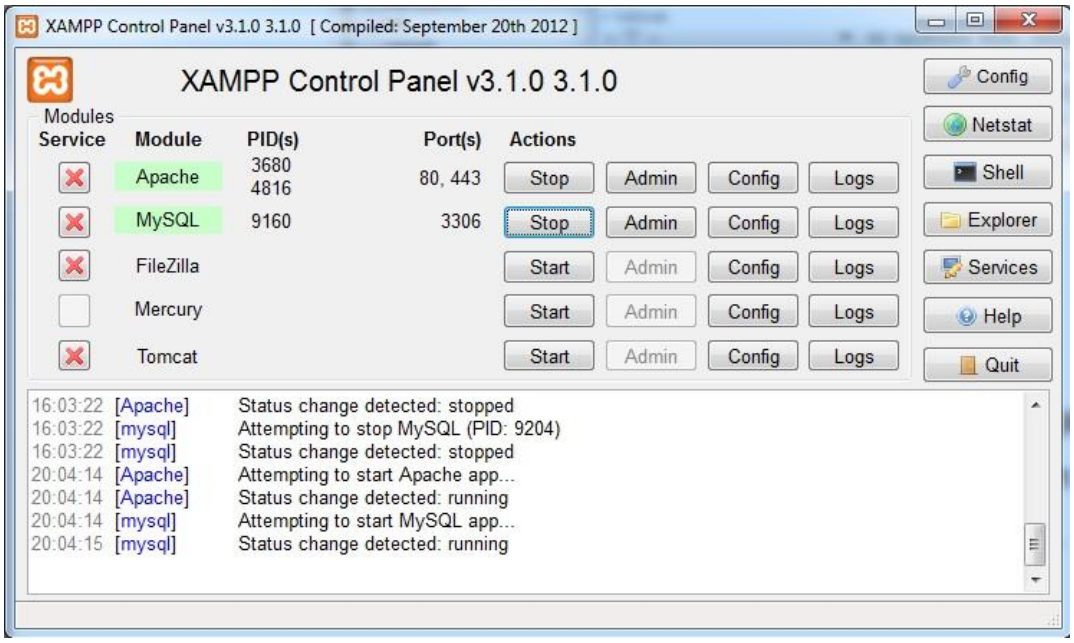
## 5.4 XAMPP

Opinnäytetyön ohjelmaa tehdessä tietokantojen ja paikallisen palvelimen tekoon käytettiin *XAMPP*ia, joka on web-palvelinpaketti. *XAMPP*in avulla saa kahta painiketta painamalla Apache-palvelimen ja MySQL:n käyttöön (painetaan Apachen ja MySQL:n kohdilla Start-painikkeita (Kuva 16).



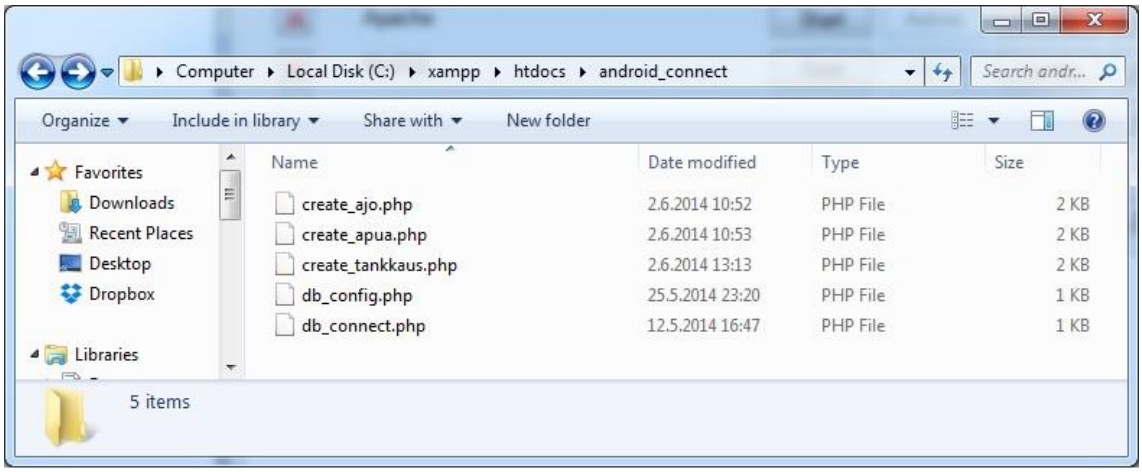

Kuva 16. XAMPP alkunäkymä

Tämän jälkeen Apache ja MySQL ovat käynnistetty (Kuva 17).




Kuva 17. XAMPP näkymä Apachen ja MySQL:n käynnistämisen jälkeen

XAMPP tukee myös PHP-tiedostoja, joiden avulla onnistuu tietojen lisääminen tietokantaan. Tässä tapauksessa, jossa opinnäytetyön ohjelman palvelimena toimi paikallinen palvelin, lisätään tarvittavat PHP-tiedostot XAMPPin asennuskansioon (tässä tapauksessa: C:\xampp\htdocs\android_connect) (Kuva 18).




Kuva 18. Ruudunkaappaus palvelimen PHP-tiedostoista

Kyseiseen kansioon viitataan Java-koodissa seuraavalla tavalla:

*http://192.168.11.3/android_connect/*

Tässä on viittaus määriteltyyn PHP-tiedostoon seuraavasti:

*http://192.168.11.3/android_connect/create_ajo.php*

*192.168.11.3* on XAMMPia pyörittävän koneen IPv4-osoite (Internet Protocol version 4 on TCP/IP-mallin Internet-kerroksen protokolla). Jos palvelin on paikallisessa verkossa, pitää Android-laitteen ja XAMMPia pyörittävän tietokoneen olla samassa verkossa, esimerkiksi samassa langattomassa verkossa. Jos opinnäytetyön ohjelmaa käytetään Android-emulaattorilla oikean puhelimen tai taulutietokoneen sijaan, pitää ohjelmointiympäristössä muuttaa *192.168.11.3* IPv4-osoite seuraavaan *10.0.2.2.*

# 6 Tekstiviestien lähettäminen Android-ohjelman avulla

Ohjelmalla voi lähettää tekstiviestin kahdella eri tavalla: ensimmäinen tapa on ohjelman aloitusruudussa painaa isoa harmaata painiketta ja kun Googlen puheentunnistus on aktivoitunut, sanotaan ohjelmalle komento *apua* (Kuva 19), jolloin ohjelma lähettää valmiiksi määriteltyyn puhelinnumeroon tekstiviestin: ”Apua apua apua” (Kuva 20).


Kuva 19. Näkymä *apua*-komennon antamisen jälkeen


Kuva 20. Kuva puhelimen ruudusta, johon on lähetetty avunpyyntöjä

Toinen tapa on sanoa ohjelmalle komento *tekstiviesti*, jolloin seuraava ikkuna avautuu (Kuva 21):


Kuva 21. Ikkuna, jonka kautta lähetetään tekstiviestejä

Kun *Anna puhelinnumero* tekstikenttää painetaan, avautuu kuvan 7. tai 9. tapainen *Puhu nyt* ikkuna, johon sanotaan puhelinnumero. Samalla tapaa toimii myös *Syötä tekstiviesti* tekstikenttä, johon sanotaan numeron sijaan viesti, joka

halutaan lähettää. Tämän jälkeen painetaan vain *Lähetä*-painiketta ja tekstiviesti lähetetään syötettyyn puhelinnumeroon.

### 6.1 Tekstiviestien lähettäminen ohjelmointiympäristössä

Ensiksi *Android Manifestiin* pitää ohjelmoida seuraavasti:

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

Tämä kohta ohjelmassa antaa ohjelmalle luvan lähettää tekstiviestejä. Ilman kyseistä lisäystä ohjelma ei saa lupaa lähettämään tekstiviestejä.

Ohjelmistoympäristön puolella tekstiviestin lähettäminen onnistuu kahdella ohjelmarivin avulla.

```
SmsManager smsManager = SmsManager.getDefault();
smsManager.sendTextMessage(numero, null, viesti, null, null);
```

Ensimmäisellä rivillä luodaan oletusinstanssi SmsManagerista, ja seuraavalla rivillä sitä käytetään hyväksi tekstiviestin lähettämiseen. Kohdissa numero ja viesti määritellään vastaanottajan puhelinnumero ja viestin sisältö String-muodossa. Tyhjiksi eli nulleiksi jäävät kohdat ovat *String scAddress, PendingIntent sentIntent* ja *PendingIntent deliveryIntent*, niitä ei tässä tapauksessa tarvita (26).

### 6.2 Vaatimukset tekstiviestien lähettämiseen

Tämä toiminto vaatii, että Android-laitteeseen on asetettu SIM-kortti (Subscriber Identity Module on yksilöllinen älykortti matkapuhelinliittymän tilaajille), jossa on toimiva matkapuhelinliittymä. Pelkällä internetyhteydellä ei voi lähettää tekstiviestejä, esimerkkinä jos ohjelmaa käyttää Android-taulutietokoneella, joka on kytkettynä langattomasti verkkoon.

## 7 Testaus

Googlen puheentunnista jouduttiin testaamaan useaan kertaan, jotta saatiin selville, kuinka hyvin puheentunnistus tunnistaa sanotut sanat ja luvut, sekä myös liittyykö puheentunnistukseen mitään rajoitteita.

Testauksessa selvisi, että puheentunnistuksen kautta tunnistetut sanat palautuvat kaikki pieninä kirjaimina. Esimerkiksi, jos Googlen puheentunnistukselle sanotaan lause: ”Koirani Musti on valkoinen”, niin palvelimelta takaisin tullut merkkijono on: ”koirani musti on valkoinen”. Tunnistettavaksi lähetetyn lauseen sanamäärälle ei testauksessa löytynyt pituusrajaa, joten lause voi olla niin pitkä kuin käyttäjä haluaa sen olevan. Sanotut välimerkit, kuten pilkut (,) ja pisteet (.) eivät näkyneet merkkijonossa pilkkuina ja pisteinä, vaan sanoina ”pilkku” tai ”piste”.

Lukujen tunnistamisessa Googlen puheentunnistuksen kanssa ilmeni paljon ongelmia. Esimerkiksi, jos Googlen puheentunnistukselle sanoi luvun ”95” (yhdeksänkymmentäviisi), niin välillä palvelin palautti merkkijonon sanana ”yhdeksänkymmentäviisi”, mutta suurimmassa osassa tapauksia numeroina ”95”. Desimaalilukuja puheentunnistus tunnisti maksimissaan kahden desimaalin verran, ja niiden sanomiseen pystyi käyttämään pilkkua tai pistettä (lukuja sanottaessa merkkijonossa pilkku(,) muuttui aina pisteeksi(.)). Desimaaliluvut toimivat vain lukuun sata (100) asti, jonka jälkeen pilkku(,) tai piste(.) merkki tunnistettiin merkkijonossa numeroiden välissä sanoina ”pilkku” tai ”piste”.

# 8 Pohdinta

Tässä luvussa kerrotaan ohjelman mahdolliset jatkokehitysmahdollisuudet, kerrotaan opinnäytetyön onnistumiset ja epäonnistumiset, sekä mitä opinnäytetyötä tehdessä opittiin.

## 8.1 Jatkokehitysmahdollisuudet

Palvelin pitäisi siirtää lähiverkosta (rajoitetulla alueella toimiva tietoliikenneverkko) internetiin. Tämä mahdollistaisi sen, että ohjelmaa voisi käyttää mistä tahansa käyttäjä haluaa, eikä sillä olisi rajoitteena langattoman verkon kantoalue.

Opinnäytetyöksi tehty ohjelma halutaan myös toimivan muilla käyttöjärjestelmillä, esimerkiksi iOS- ja (Applen kehittämä käyttöjärjestelmä,

joka on käytössä Applen iPhone-, iPod touch-, iPad- ja Apple TV – laitteissa) Windows Phone 8 käyttöjärjestelmissä(Microsoftin tekemä mobiilikäyttöjärjestelmä).

Android-ohjelmalle halutaan myös luoda käyttäjätilijärjestelmä, tämä mahdollistaisi käyttäjien yksilöinnin tietokannassa.

Opinnäytetyön alkuperäinen idea oli luoda Saimaan ammattikorkeakoulun sosiaali- ja terveysalalle ohjelma, jonka avulla olisi voitu kirjata arkielämän toimenpiteitä puheohjatusti tietokantaan, josta koehenkilön arkielämän toimenpiteitä olisi voitu seurata. Näihin seurattuihin toimenpiteisiin olisi kuulunut esimerkiksi heräämisen kirjaus, kaupassa käynnit, ruokailuajat, liikunta, tietokoneen käyttö, television katsominen ja nukkumaanmeno. Nykyisestä ohjelmasta olisi helppo pienillä muutoksilla muokata yllämainittujen vaatimusten sisältävä ohjelma.

Yksi mahdollisista kehitysmahdollisuuksista olisi ollut tehdä ohjelma, joka olisi jatkuvasti päällä laitteella eli laitteen näyttöä ei suljettaisi ja ohjelma pyörisi laitteella aktiivisesti. Ohjelman käyttöliittymänä toimisi yksi iso painike, jonka avulla annettaisiin komentoja tai vaihtoehtoisesti puheentunnistus olisi koko ajan päällä. Tämä olisi kotona, sairaalassa tai vanhainkodeissa asuville vanhuksille suunnattu ohjelma. Yksi esimerkki tapauksista voisi olla, että käyttäjä antaa ohjelmalle komennon *Kaisa*. Kaisa tässä tapauksessa olisi käyttäjän tytär, ohjelma tunnistaa komennon *Kaisa* ja soittaa käyttäjän lapselle. Tai, jos ohjelmalle syöttää puheella komennon *apua,* se lähettäisi esimerkiksi lapsille tai hoitohenkilökunnalle viestin, että vanhus tarvitsisi apua ja samalla kertoisi tämän olinpaikan. Laitteena näissä tapauksissa voisi toimia esimerkiksi iso tabletti, joka olisi helppo vaikka laittaa seinälle tai pitää näkyvällä paikalla huoneistossa. Tulevaisuudessa, jos esimerkiksi älykellot yleistyvät, olisi ohjelma helppo tehdä kyseiseen laitteeseen. Tällöin ohjelma olisi aina käyttäjän mukana ja olisi nopeasti käytettävissä.

## 8.2 Onnistumiset ja epäonnistumiset

Nämä kaikki halutut asiat onnistuttiin tekemään: ohjelmalla pystyy antamaan puhekomentoja, lisäämään tekstikenttiin tietoja puheella, annetut tiedot lisätään tietokantaan ja ohjelmalla voidaan lähettää tekstiviestejä.

Ainoa asia, missä epäonnistuttiin, oli tekstin muuttaminen puheeksi, jonka avulla ohjelma olisi ilmoittanut suomen kielellä käyttäjälle esimerkiksi, milloin puheentunnistus on toiminnassa, lähtikö avunpyyntö eteenpäin ja onnistuiko tietojen lisääminen tietokantaan. Ohjelma saatiin puhumaan, mutta puhuttu kieli oli niin sekavaa, että siitä ei saanut kunnolla selvää. Jotta puhutusta kielestä olisi saanut selvää, olisi ohjelmaan pitänyt lisätä jonkinlainen lisäosa, joka parantaisi puhutun kielen selkeyttä eikä se olisi niin tietokonemaista.

## 8.3 Mitä opin tehdessä opinnäytetyötä

Opinnäytetyön ansioista osaan luoda Android-käyttöjärjestelmälle ohjelmia, koodata Javaa, luoda ulkoisia tietokantayhteyksiä Javan avulla, käyttää Googlen puheentunnistusta eri tavoin hyödyksi ohjelmissa, tehdä PHP-tiedostoja tietokantoja varten, muokata käyttöliittymää XML:n avulla, muuntaa Javalla tekstiä puheeksi, sekä tehdä HTTP-pyyntöjä. Tiedon etsintä ja tiedon muuntaminen omiin tarkoituksiin on parantunut, ja opin myös paljon virheiden ratkaisemisesta.

## Kuvat

## Lähteet


1. Developers. Activity.
http://developer.android.com/reference/android/app/Activity.html.

2. Lars, V. Android Development – Tutorial. Android Manifest.
http://www.vogella.com/tutorials/Android/article.html.

3. Wikipedia. Apache HTTP Server.
http://fi.wikipedia.org/wiki/Apache_(palvelinohjelma).

4. Wikipedia. Apple. http://fi.wikipedia.org/wiki/Apple.

5. Developers. AsyncTask.
http://developer.android.com/reference/android/os/AsyncTask.html.

6. Pieraccini, R. 2012. The Voice in the Machine, 55-58.

7. Pieraccini, R. 2012. The Voice in the Machine, 90-92.

8. Developers. doInBackGround.
http://developer.android.com/reference/android/os/AsyncTask.html.

9. Pieraccini, R. 2012. The Voice in the Machine, 212-213.

10. Pieraccini, R. 2012. The Voice in the Machine, 212-213.

11. Wikipedia. Eclipse (IDE). http://fi.wikipedia.org/wiki/Eclipse_(IDE).

12. Pieraccini, R. 2012. The Voice in the Machine, 120-124.

13. AndroidHive. Android making HTTP Requests.
http://www.androidhive.info/2011/10/android-making-http-requests/

14. Opetusvirasto. If-lause. http://ideaport.edu.hel.fi/C/Ckieli/sivuc5.html.

15. Developers. Intent.
http://developer.android.com/reference/android/content/Intent.html.

16. Wikipedia. IPv4. http://fi.wikipedia.org/wiki/IPv4.

17. Helsingin yliopisto. Metodit.
http://www.cs.helsinki.fi/u/ahslaaks/kesa11/ohpe/metodit.html.

18. Wikipedia. MySQL. http://en.wikipedia.org/wiki/MySQL.

19. Wikipedia. Parametri. http://fi.wikipedia.org/wiki/Parametri_(tietotekniikka).

20. Wikipedia. Merkkijono. http://fi.wikipedia.org/wiki/Merkkijono.

21. Microsoft. Try-catch. http://msdn.microsoft.com/en-us/library/0yd65esw.aspx.

22. Wikipedia. URl. http://fi.wikipedia.org/wiki/URI.

23. Wikipedia. XAMPP. http://fi.wikipedia.org/wiki/XAMPP

24. Pinola, M. 2011. Speech Recognition Through the Decades. http://www.techhive.com/article/243060/speech_recognition_through_the_decades_how_we_ended_up_with_siri.html?page=2.

25. Pinola, M. 2011. Speech Recognition Through the Decades. http://www.techhive.com/article/243060/speech_recognition_through_the_decades_how_we_ended_up_with_siri.html?page=2

26. Developers. SMSManager. http://developer.android.com/reference/android/telephony/SmsManager.html.

# Liitteet

## Syventävää tietoa

### JSON (AsyncTask selitetty)

JSON on Java-luokka ohjelmointiympäristön puolella, jonka avulla voimme lähettää ohjelman keräämät tiedot palvelimen PHP-tiedostojen kautta ulkoiseen tietokantaan.

Tietojen lähettämiseen tarvitsemme kaksi asiaa: PHP-tiedoston URL-osoitteen ja parametrit jotka lähetetään eteenpäin. URL on lyhenne sanoista Uniform Resource Identifie ja tarkoittaa merkkijonoa, jolla kerrotaan halutun tiedon paikka.

URL-osoite voidaan määrittää ohjelmointiympäristössä seuraavalla tavalla:

```
private static String url_create_tallenne = "http://192.168.43.240/android_connect/create_tankkaus.php";
```

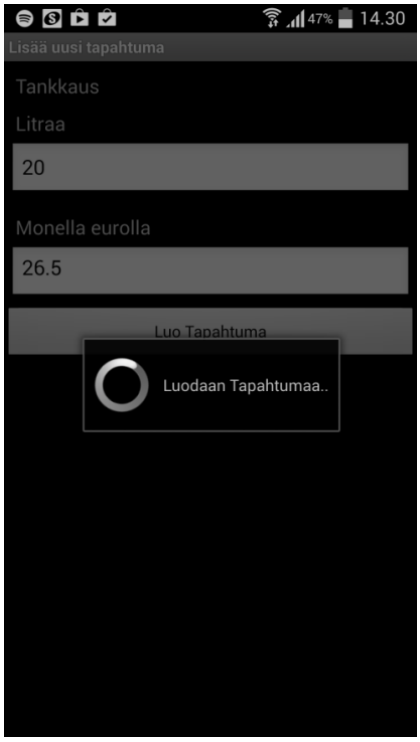
Tämä on URL-osoite on tie tankkaamisen tietojen lähettämistä varten olevaan PHP-tiedostoon, muiden tietojen lähettämistä varte on omat URL-osoitteet.

Seuraavaksi kerätään tiedot ohjelman tekstikentistä lähettämistä varten:

```
inputTapahtuma = (EditText) findViewById(R.id.inputTapahtuma);
inputMaara = (EditText) findViewById(R.id.inputMaara);
inputEuroa = (EditText) findViewById(R.id.inputEuroa);
```

Loput ohjelmoinnista tehdään AsyncTaskin alla. AsyncTask on abstrakti luokka, joka mahdollistaa suorittamaan operaatioita taustalla ja näyttämään tulokset käyttöliittymän puolella. Eli tässä tapauksessa tietojen lähettämiseen tietokantaan saattaa mennä useampi sekuntti (tämä riippuu palvelimen- ja käytettävän laitteen verkon nopeudesta), kun ohjelmassa painetaan *Luo Tapahtuma* painiketta ilmestyy ruudulla ponnahdusikkuna, jossa lukee Luodaan Tapahtumaa. Ja ponnahdusikkuna sulkeutuu, kun tiedot on saatu lähetettyä tietokantaan.

Seuraavat tietokantaan lähettämiseen tarvittavat ohjelmakoodit käydään läpi sillä aikaa, kun Luodaan Tapahtumaa ponnahdusikkuna on aktiivisena. Tätä varten luodaan uusi luokka, joka perii AsyncTaskin. Ponnahdusikkuna on tehty käyttämällä ProgressDialogia, joka on ikkuna mikä näyttää, että jotain yritetään ladata ohjelman taustalla ja se sulkeutuu kun lataus on valmis (Kuva 1).




Kuva 1. Progress Dialog näkyy, kun tietoja ollaan lisäämässä tietokantaan

```
class CreateNewTallenne extends AsyncTask<String, String,
String> {

@Override
protected void onPreExecute() {
super.onPreExecute();
pDialog = new ProgressDialog(NewTapahtumaActivity.this);
pDialog.setMessage("Luodaan Tapahtumaa..");
pDialog.setIndeterminate(false);
pDialog.setCancelable(true);
pDialog.show();
}
```

Muutetaan saadut tiedot String-muotoon:

```
String tapahtuma = inputTapahtuma.getText().toString();
String maara = inputMaara.getText().toString();
String euroa = inputEuroa.getText().toString();
```

Sitten rakennetaan parametrit, eli yhdistetään kaikki tiedot samaan taulukkoon:

```
List<NameValuePair> params = new ArrayList<NameValuePair>();
params.add(new BasicNameValuePair("tapahtuma", tapahtuma));
params.add(new BasicNameValuePair("maara", maara));
params.add(new BasicNameValuePair("euroa", euroa));
```

Lopuksi teemme JSON-luokalle HTTP-pyynnön, joka sisältää: tankkauksen PHP-tiedoston URL-osoitteen.

```
JSONObject json = json-
Parser.makeHttpRequest(url_create_tallenne,
                  "POST", params);
```

Tiedot vastaanotetaan alla olevalla ohjelmointikoodilla JSON Java-luokassa. Ja luodaan uusi BackGroundTask käyttämällä saatuja tietoja.

```
public JSONObject makeHttpRequest(String url, String method,
                  List<NameValuePair> params) {
BackGroundTask Task= new BackGroundTask(url, method,  params);
}
```

Lopuksi luodaan JSON Java-luokkaan public class BackGroundTask, jonka avulla lähetämme tiedot palvelimen PHP-tiedostoihin.

```
public class BackGroundTask extends AsyncTask<String, String,
JSONObject>{

List<NameValuePair> postparams= new ArrayList<NameValuePair>();
String URL=null;
String method = null;
```

Aluksi tarvittavat muuttujat alustetaan.

```
public BackGroundTask(String url, String method,
List<NameValuePair> params) {

URL=url;
postparams=params;
this.method = method;
          }
```

Ja tämän jälkeen itse tietojen lähettäminen palvelimen PHP-tiedostoihin tehdään doInBackground-metodia käyttäen. DoInbackGround on yksi

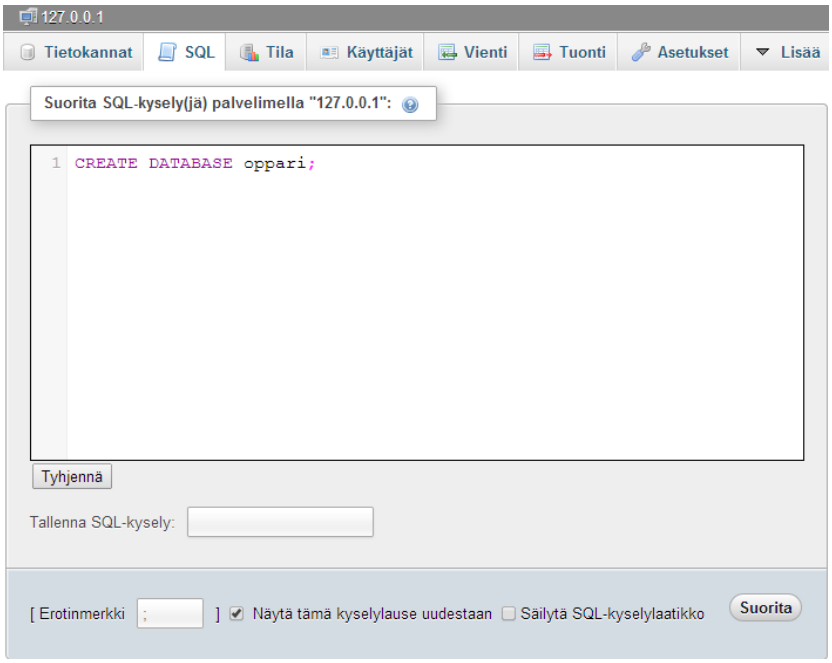
AsyncTaskin askeleista, jonka aikana suoritetaan pitkään pyöriviä operaatioita. AsyncTask on puolestaan abstrakti luokka (luokka on abstrakti, jos se ei toteuta kaikkia metodeitaan), joka mahdollistaa suorittamaan operaatioita taustalla ja näyttämään tulokset käyttöliittymän puolella.

```
@Override
protected JSONObject doInBackground(String... params) {

if(method == "POST"){

DefaultHttpClient httpClient = new DefaultHttpClient();
HttpPost httpPost = new HttpPost(URL);
httpPost.setEntity(new UrlEncodedFormEntity(postparams));
HttpResponse httpResponse = httpClient.execute(httpPost);
```

Ensiksi luodaan HTTP-client ja HTTP-POST (tämän avulla lähetetään tietoa www-palvelimelle), tämän jälkeen ennen HTTP-requestin (lähettää palvelimelle pyynnön) tekemistä pitää kaikki lähetettävä data koodata, jotta merkkijono data saadaan muutettua URL-osoitteelle tarkoitettuun muotoon. Ja lopulta suoritetaan HTTP-POST käyttämällä HTTP-clienttiä. Ja HTTP-Responsen avulla haetaan vastaus palvelimelta toimiko tietojen lähettäminen tietokantaan.

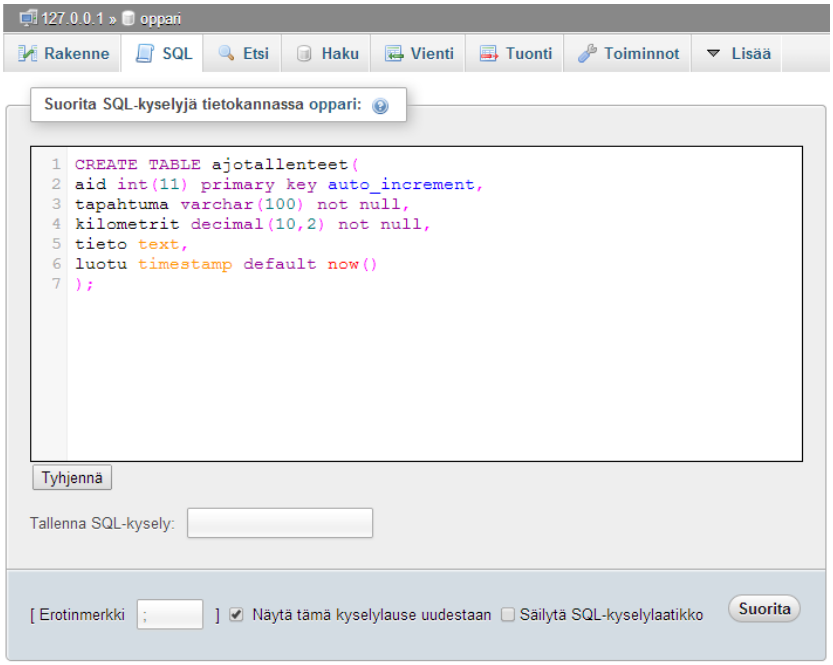
**phpMyAdmin tarkennettu tietokantojen luonti**

Tietokanta ja taulut luotiin käyttämällä phpMyAdminista löytyvää SQL-toimintoa. PhpMyAdminin etusivulta painetaan SQL-saraketta, jotta phpMyAdminiin saadaan syötettyä SQL-lauseita. Sitten vain kirjoitetaan tesktikenttään haluttu SQL-lause, tässä tapauksessa *CREATE DATABASE oppari;* , jonka avulla luodaan tietokanta nimeltä *oppari* ja painetaan *suorita*-painiketta (Kuva 2).




Kuva 2. Tietokannan luominen phpMyAdminissa SQL:n avulla

Kun tietokanta on saatu luotua, avataan sen etusivulta painamalla luodun tietokannan nimeä. Tämän jälkeen painetaan taas SQL-saraketta, jotta kyseiselle tietokannalle voidaan suorittaa SQL-kysely. Alla olevan kuvan SQL-kyselyllä luodaan taulukko nimeltä *ajotallenteet*, joka sisältää 5 saraketta. Tähän taulukkoon merkitään ajopäiväkirjan tallenteet.




Kuva 3. Taulun luominen tietokantaan phpMyAdminissa SQL:n avulla

Ensimmäinen sarake on *aid* (ajo id)(Kuva 3), joka on maksimiltaan 11 merkkiä pitkä kokonaisluku, se on myös pääavain ja se täytetään automaattisesti. Automaattisella täytöllä tarkoitetaan, kun taulukkoon lisätään tietoa saa ensimmäinen tietue ajo id:ksi 1, seuraava 2 ja niin edelleen.

Seuraava sarake on tapahtuma, joka on *varchar(100), not null*. *Varchar(100)* tarkoittaa, että kohtaan voi kirjoittaa 100 merkkiä pitkän merkkijonon. *Not null* tarkoittaa, että kohta ei saa olla tyhjä, kun tietokantaan lisätään tietoa.

Kolmas sarake on kilometrit, joka on *decimal(10,2)*, not null. *Decimal(10,2)* kohtaan voi lisätä numeerista dataa, jossa on kaksi desimaalia.

Neljäs tieto on *text*, eli siihen voi kirjoittaa tekstiä rajattomasti.

Viides kohta on *luotu*, mikä *on timestamp default now()*. Eli kun tietue lisätään tietokantaan, lisätään perään kellonaika ja päivämäärä, jolloin tieto on lisätty tietokantaan.

**Liitteen syventävää tietoa kuvat**